retrieve [a] one-dimensional [array] slices of pixels with a length equal to the width of the portion; and

determine for [each of] the [one-dimensional array] slices of pixels whether the [array has an] slices have intensity minima, by measuring a distance from the center of an assumed intensity minima out to a dominant background color for each of said slices.

(Amended) ~~41~~. The computer program product of claim ~~40~~ wherein for any slices that contain no dominant background color pixels in the appropriate direction such [samples] slices are considered to be invalid and are discarded.

(Amended) ~~42~~. The computer program product of claim ~~41~~ wherein for those [samples] slices that have a dominant background color at [the] an appropriate location, a center for the crease is determined by averaging intensity at the centers of [the best] valid slices.

(Amended) ~~43~~. The computer program product of claim ~~42~~ wherein the first average of the centers of all the valid slices are sorted by increasing distance from the first average and the average is recomputed using only the centers of the highest (NSLICES/2)+1, where (NSLICES) is the number of slices.

(Amended) ~~44~~. The computer program product of claim ~~43~~ wherein the whole area is considered to be invalid if there are less than (NSLICES/2)+1 valid slices.

(Amended) ~~45~~. The computer program product of claim ~~44~~ wherein a composite width is assigned for the area crease as the minimum area slice width, and a composite vector of intensities for each

slice is constructed from the center point of the crease to the near dominant background color point for the slice.

(Amended) ~~46~~ 7. The computer program product of claim ~~45~~ 6 wherein an array corresponding to the composite vector of intensities for each slice is filled in as follows:

for a "center" area;

define slice(I) to be the pixel in a slice that is I number of pixels from the center in the direction of the near-DBC point; and

fill in the array;

array[i] = average of intensities of the slice[i] pixels for the valid slices; and

iterate over I from the center out to the near-DBC point as:

array[i] = maximum of array[i] and array[i - 1] for each side of the crease, producing two arrays.

(Amended) ~~47~~ 45. The computer program product of claim ~~45~~ 43 further comprising the step of;

assigning a quality to each area of the page with the quality being equal to the width of the crease found or an invalid crease indicator if the area/crease fails to qualify as a crease

if there are less than (NSLICES/2)+1 valid slices, or the width is below a minimum crease width, or if the majority of centerpoints used to construct the average centerpoint are not within a constant horizontal distance or one another or if the vector of intensities ~~appears~~ concave.

(Amended) ~~48~~ 9. The computer program product of claim ~~47~~ 8 wherein the crease with the highest quality is determined as the crease

for the page.

(Amended) ~~49~~ 47. A computer program product for removing a crease stored on a computer readable media, comprises instructions for causing a computer to:

set all pixels to the outside of the center portion of a left or right side of the image crease to a dominant background color.

(Amended) ~~51~~ 49. The computer program of claim ~~50~~ 48 wherein the instruction for causing the computer to bleach comprises instructions for causing the computer to:

define array{i} to [define array[i] to] be the intensity in the creases's intensity vector at a distance I pixels from the center;

define image {y}{x} to [define image[y][x] to] be the pixel in the image x pixels horizontally and y pixels vertically from the upper-left corner;

define center to be the center of the crease and width to be its width;

define intensity(pixel) to be a function that returns the intensity of a pixel;

for a left-side crease, iterate over y, for each row in the image, iterate over I from a fixed distance over crease width:

if ((intensity(image{y}[center + I] + (intensity(DBC) - array{i}))) > (0.90 * intensity(DBC)))

set image{y}[center + I] to DBC and set the corresponding pixels in the B/W image to white

[if ((intensity(image[y][center + I] + (intensity(DBC) -